1970-12

AUTO: an automation simulator

Gold, Bennett Alan

Monterey, California. Naval Postgraduate School

AUTO: AN AUTOMATON SIMULATOR

by

Bennett Alan Gold

United States
Naval Postgraduate School

THESIS

AUTO: AN AUTOMATON SIMULATOR

by

Bennett Alan Gold

December 1970

AUTO: An Automaton Simulator

by

Bennett Alan Gold
Lieutenant Commander, United States Navy
B.A., Stanford University, 1960

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1970

ABSTRACT

AUTO: An Automaton Simulator, a program implemented under IBM 360/67 TSS, is capable of simulating any user-defined deterministic or nondeterministic finite automaton, pushdown automaton, Turing machine or procedural Turing machine. The simulation is achieved through the use of two pushdown stacks and a single finite control. This approach provides efficient simulation and avoids the programming duplication required to simulate each type of automata separately. Interactive capabilities make the system particularly well suited to the online design, modification, and testing of user-defined automata.

TABLE OF CONTENTS

I. INTRODUCTION

The objective of this research was to devise an aid for the teaching of formal languages and automata theory. The system developed allows the student to design, test, and change automata in an interactive manner. This process permits the user to observe the step by step operation of a defined automaton and if desired, to correct or alter its operation. This eliminates the need for lengthy and tedious hand simulations.

AUTO: An Automaton Simulator, can simulate the operation of deterministic and nondeterministic finite automata, pushdown automata, Turing machines, and procedural Turing machines. Since the system is capable of simulating several different types of automata with only one interface, it eliminates the duplication of effort required by separate simulations of each type of automata.

II. SYSTEM DESCRIPTION

AUTO is a deterministic automaton which can, by performing transformations on the rules of any given automaton, simulate that automaton. Since each procedure performed by any automaton can be performed by a Turing machine, then theoretically it is only necessary to simulate a Turing machine in order to have the capability of simulating any automaton. The simulation of a Turing machine can be done by a deterministic, two pushdown machine. It is easily seen that those symbols to the left of the head of the Turing machine being simulated can be stored on one pushdown list, while the symbols on the right of the head can be placed on the other pushdown list. AUTO, using two pushdowns, can also simulate other types of automata more effectively then the Turing machine, since when conducting a simulation, Turing machines often carry out their computations most inefficiently.

AUTO is designed so that at all times the two pushdown simulator remains invisible to the user. The user is unaware that he is using a simulator, but feels rather that he is interacting directly with the automaton which he has created. This feature is maintained during the input, operation, and output phases of the simulation. The addition of new interfaces or modification of the present ones would allow the simulation of other types of automata, such as two-way tape Turing machines or the two-stack pushdown automata, of which AUTO is an example.

A. THE SIMULATION

AUTO is capable of simulating any given automaton by the use of two pushdown stacks and a single finite control. The internal transformational rules upon which AUTO operates have a canonical form distinct from that of any other automaton. The canonical form of the transformational rules by which AUTO operates is of the form

$$(Q_i, L_i, R_i) - (Q_j L_j, R_j) \quad \text{where}$$

Q_i is a state of finite control.

L_i is the top symbol of the left pushdown.

R_i is the top symbol of the right pushdown.

Q_j is the new state of finite control.

L_j is the symbol(s) which replace L_i.

R_j is the symbol which replaces R_i.

The left hand side of the rule represents an initial configuration and the right hand side, the configuration after the indicated transformation is performed.

Since none of the machines or systems under consideration have rules exactly of this form, all input rules must first be converted into the above canonical form for AUTO. Those items of the above configuration which have no exact counterpart in the rules of the system being simulated are filled in with dummy arguments by AUTO. These arguments vary according to the system which is being simulated and will indicate to AUTO the proper action to take in the simulation. An example of a rule for a finite automaton might be

$$(Q_1, a) - (Q_2).$$

The resulting rule in AUTO would be

$$(Q_1, \#, a) - (Q_2, \#, \&)$$

which would indicate that the state of the finite control is changed from Q_1 to Q_2; the top of the left pushdown which is ignored for a finite automaton is left unchanged; and the symbol on the top of the right pushdown is popped.

Similarly, transformations are made upon the rules of the pushdown automaton, Turing machine, and procedural Turing machine to conform to the canonical format used by AUTO. In so doing, the system preempts the use of certain symbols for system usage. The symbols, '#', '%', and '&' may not be used as symbols of the user's input or pushdown alphabet. The '&' may be used however, in any transformational rule when reference is made to the null or empty string.

B. THE SIMULATION

All simulations have an identical initialization procedure. First, delimiters are placed on top of both stacks for use as end of string, end of pushdown, or end of tape symbols. The input string is then placed on top of the right stack and the finite control is set to the start state. Upon completion of this initialization, the procedure will vary according to the type of machine under consideration.

1. The Finite Automation (FA)

The left stack is not used in this simulation. Instead of a left to right scan of the normal FA, AUTO does a top to bottom scan, by continually popping off the top symbol of the input stack (i.e., right pushdown), while following the general procedural rules for

the operation of a finite automaton. Upon completion of processing of the last input character, the finite control will be in some state, Q_j, which is then compared against a list of final states. On the basis of this test, the input string will either be accepted or rejected.

2. The Pushdown Automaton (PDA)

There are two types of pushdown automata; the final state PDA and the empty store PDA. AUTO is capable of simulating either of these types. The start symbol is placed on top of the left stack. Input symbols are popped one at a time from the right stack while the action indicated by the rules is performed upon the left pushdown. An exception to this procedure is the case of epsilon rules, which permit the finite control to change state and the left pushdown to be manipulated without popping the top of the input stack. The problem of avoiding an infinite loop when processing a PDA with epsilon rules is discussed in a later section dealing with halting the simulation.

In all cases where complete processing of the input string is attained, i.e., the initialization delimiter is encountered, the next action will be determined by the type of PDA being simulated. For the final state PDA, action similar to that of the FA will be taken. If the automaton is an empty store PDA, the top of the left stack is examined. If the top symbol is the delimiter originally placed on the stack during initialization, then the pushdown is considered to be empty and hence the string would be accepted, otherwise the string would be rejected.

3. The Turing Machine (TM)

The input tape for the TM is not followed by a delimiter, but rather by an unbounded string of blanks, which are added as needed as the tape head moves to the right. AUTO always considers the tape head to be at the top of the right (input) stack. For a move right, the top of the right stack is transformed according to the applicable rule, then popped off the right stack and pushed on top of the left pushdown. For a left move, the process is reversed so that the top of the left stack is popped and then pushed down on top of the symbol on the right stack which had just been processed. Before a left move is attempted, the top symbol of the left pushdown is examined and if it is the initialization delimiter, then a move left would be equivalent to having the tape head move off the end of the tape. Since this is not permissable in a TM, the input string would be rejected.

After each move has been processed, a check is made for final state acceptance. If a final state has not been reached the processing continues until either a final state is obtained or a particular configuration is reached for which there is no applicable rule. This latter case would terminate processing and cause the input string to be rejected.

4. The Procedural Turing Machine (PTM)

The PTM functions according to the same procedures as the TM except that the PTM does not accept or reject input strings. Rather the PTM continues to operate until there are no rules which apply to a given configuration and then the tape at that stage will be output as the result of the procedure. An example is the PTM demonstrated in the SAMPLE TERMINAL SESSION. The rules of this PTM

define unary multiplication. The number 3, represented as 111, is multiplied by 2, represented as 11. The input is defined as the string 111*11* and the final output is XXX*11*111111, where 111111 represents the answer 6.

C. NONDETERMINISTIC AUTOMATA

The nondeterministic automaton can be represented as a tree-like structure where each node represents a particular configuration (lefthand side of a canonical rule) and the edges represent the allowable transformations from one configuration to another. If any path terminates in an acceptable final configuration, then the input string is accepted. To avoid a fruitless search for such a path, the tree is developed in breath first, then in depth. This approach determines the shortest possible acceptance path, as well as avoiding a possible infinite loop along some particular path.

The program starts with the input string, start state, and start symbol, if any, making up the initial configuration for the root node. The listing of the rules is searched for rules which are applicable; the transformation is then performed and a node allocated for each such rule. The nodes are developed and allocated, conceptually, from left to right. Each new node contains a pointer to its predecessor node and at each level every node has a pointer to the next node developed at that level. The last (rightmost) node at each level contains a pointer to the first (leftmost) node of the next level. The following example will illustrate the development of the tree.

The nondeterministic finite automaton picture in figure 1 accepts the set of all sentences containing either two consecutive 0's or two

consecutive 1's. The tree developed by this automaton while processing the input string, 0100, is shown in figure 2.

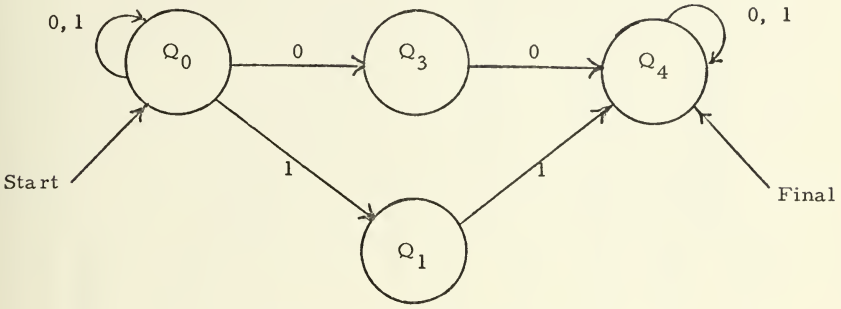


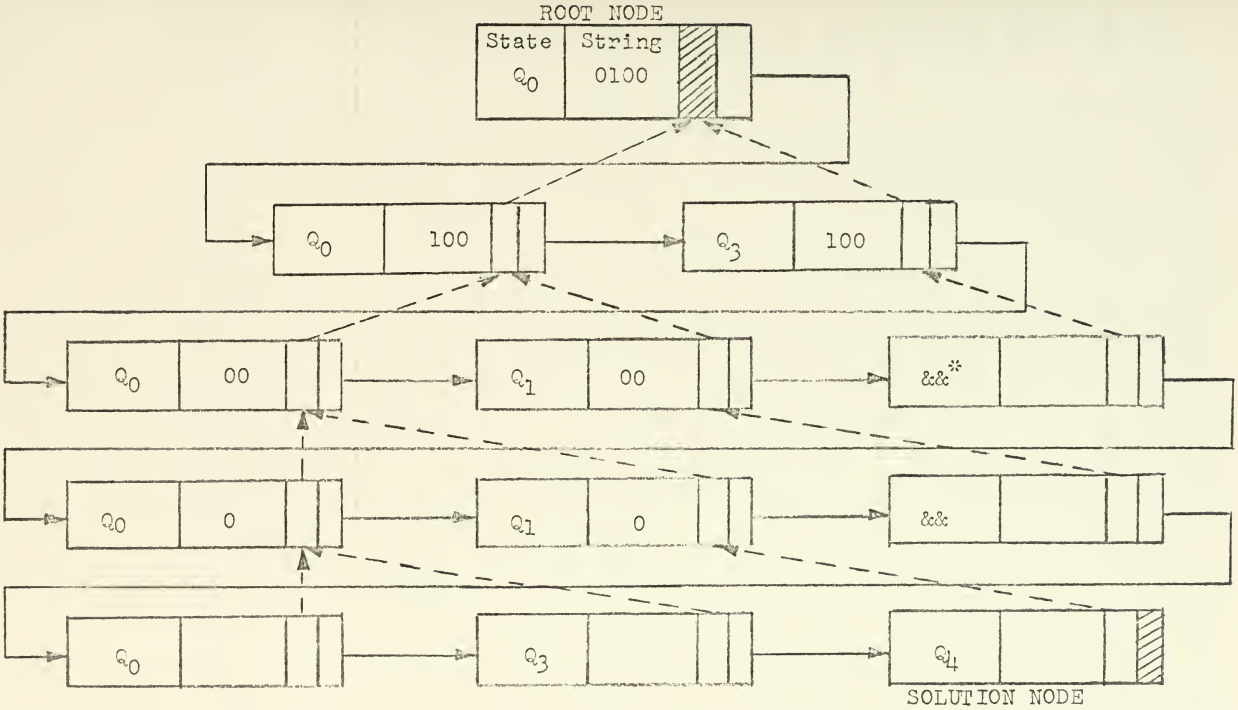
Figure 1. State Diagram

As each node is created a check is made to determine if a condition of acceptance has been reached. If so, the tree is traversed from the solution node to the root by following the predecessor pointers (indicated by --> in figure 2). As the tree is traversed, the predecessor pointers are reversed so that when the root node is reached there will be a direct path to the solution node.

Whereas, for deterministic automata, the printout is given as each step is performed, this is not possible when dealing with nondeterministic automata, since it is unknown at the time a node is created whether or not it lies on the solution path. Therefore, the user may select one of three output options: no trace, a trace only if the string is accepted, or a printout of each node as it is allocated.

D. HALTING THE SIMULATION

An important consideration in the operation of AUTO is the determination of when to halt the processing of any particular string. The problem differs for deterministic and nondeterministic automaton.



* && indicates that the predecessor node was unable to apply any rules and a final state had not been reached.

Figure 2. Tree Diagram

For the deterministic finite automaton and pushdown automaton without epsilon rules, any input string is processed until there is no applicable rule which matches the configuration, or until the end of the string has been reached. For the deterministic TM, PTM, or PDA with epsilon rules it is possible to have rules which cause the machine to go into an infinite loop. For the TM or the PTM, the tape head may move left and right alternatively without ever reaching the end of the tape or running out of applicable rules. The PDA with epsilon rules could add symbols indefinitely to the left stack without ever popping any symbols from the input stack. Therefore, AUTO requires the user to set a maximum number of allowable steps to be performed by an deterministic automaton.

The halting situation for the nondeterministic TM, PTM, or PDA with epsilon rules is basically the same as for the deterministic case. However, for all nondeterministic automata the tree structure grows in breadth rapidly in relation to the number of rules which are nondeterministic. Since it is quite possible for the number of nodes to increase almost without bound even at a fairly shallow depth in the tree, the user must declare the maximum number of nodes to be allocated.

If the number of steps or nodes set by the user proves to be insufficient in a given case, AUTO permits the user to increase the maximum and have the system continue with the simulation.

III. CONCLUSIONS

AUTO has effectively simulated deterministic and nondeterministic finite automata, pushdown automata, Turing machines, and procedural Turing machines. During all phases of interaction, the system acts as the machine defined by the user. The various options and capabilities of AUTO are demonstrated in the sample terminal session following Appendix A.

Automata students introduced to AUTO reported that the interactive system acted as a valuable learning aid. The students found that they could quickly have their automata performing as desired, because they were able to watch the step by step operation of an automaton, easily change its definition, and retest input strings.

A modified version of AUTO is feasible which would be suitable for batch operation. Although the interactive features would be sacrificed, this version would allow for quick testing and printout of several automata. After reviewing the output, the user could log onto the interactive version of AUTO, make corrections and retest the automata.

As presently designed, AUTO could be a valuable aid in the study of automata theory and in conjunction with a batch processing version, could provide the user with a versatile system for designing and testing automata.

APPENDIX A

USER'S MANUAL FOR AUTO

To access AUTO, the user logs on to the time sharing system in the usual manner and then executes the command, 'AUTO'. The system responses, permissible user responses, and default options are given below (System responses are given in UPPER CASE). All questions followed by '(Y OR N)' imply the user is to enter the single character 'y' for 'yes' or 'n' for 'no'.

1. TYPE OF MACHINE?

Response	Meaning
fa	finite automaton
pda	pushdown automaton
tm	Turing machine
ptm	procedural Turing machine
fin	to terminate session

No default.

2. IS MACHINE DETERMINISTIC? (Y OR N)
No default.

3. MAX NUMBER OF NODES?
For nondeterministic automata, enter the maximum number of nodes to be allocated. Provisions are made for revising this number later in the session. Reply must be in the range: 0 - 9999.

4. MAX NUMBER OF STEPS?
Same as above except that this is the maximum number of steps to be executed by deterministic automata.

5. GIVE RULES
The system will key the user to input rules one at a time, by printing a rule number. Acceptable rule formats are:

A. Finite Automata $(Q_i, A) - (Q_j)$ where

Q_i, Q_j are states of the finite control,
A_i is the character of the input string being scanned.

B. Pushdown Automata: $(Q_i, A, X) - (Q_j, S)$ where

Q_i, Q_j are states of the finite control,
A_i is the character of the input string being scanned, or the character '&', denoting an epsilon rule,
X is the top symbol on the pushdown stack,
S is a string of from 1 to 4 characters of the pushdown alphabet to be added to the pushdown stack, or the character '&' which denotes that the top symbol of the pushdown stack is to be popped.

C. Turing Machines and Procedural Turing Machines:
$(Q_i, A) - (Q_j, Y, M)$ where

Q_i, Q_j are states of the finite control,
A_i is the character being scanned by the tape head. The symbol may be '&' at the righthand end of the tape,
Y is the single character from either the input or tape alphabet which replaces 'A',
M which is either the single character 'l' or 'r', which denotes whether the tape head is to move left(l) or right (r).

The user terminates the input of rules by typing 'end' as the last rule. States may be any string of characters up to three in length.

All parenthesis and blanks are ignored by the system and may be included as desired by the user. The left and right hand side of the rules must be separated by the symbol, '-', and the rule must contain the proper number of commas. For example:

$$(Q_i, A, X) - (Q_j, S)$$

$$Q_i, A, X - Q_j, S$$

$$((Q_i, A), (X)) - (Q_j, S)$$

are all equivalent.

6. ENTER START STATE
Same rules as for states in the input rules.
No default.

7. ENTER START SYMBOL
For PDA only. User must input a single symbol for the top of the pushdown stack.
No default.

8. FINAL STATE OR EMPTY STORE? (F OR E)
For PDA only. Enter:
'f' for final state acceptance,
'e' for empty store acceptance.
No default.

9. ENTER FINAL STATE(S)
For all final state machines, the user must enter the final state or states, one per line. End input by typing 'end'.
No default.

10. MAKE ANY CORRECTIONS? (Y OR N)
If user responds 'y', he will be given the following options:

CORRECT ANY RULES? (Y OR N)
If 'y' then system will respond:

GIVE RULE NUMBER
Enter the number of the rule to be corrected. The system will print the rule as originally written and prompt the user with:

ENTER CORRECT RULE:
Same rules as for GIVE RULES.

ADD ANY RULES? (Y OR N)
If 'y' the system will give next sequential rule number and user will respond as for GIVE RULES.

CHANGE START STATE? (Y OR N)
If 'y' then user will be prompted with:

ENTER START STATE
Rule 6 above applies.

CHANGE FINAL STATE(S)? (Y OR N)
If 'y' then the user will be prompted with:

ENTER FINAL STATE(S)
User must respond with all final states, not just new ones.

CHANGE MAX NODES OR STEPS? (Y OR N)
If 'y' then rules 3 and 4 above, apply.

11. GIVE INPUT STRING
User reply is any string of characters, without embedded blanks, or the reserved symbols: '#', '&', and '%'.

12. WANT A TRACE? (Y, N, OR A)

A. 'y' (1) A trace of all steps of a deterministic automata is printed.
(2) A trace of the solution path, if any, of a nondeterministic automata is printed.

B. 'n' No trace. Output will be a statement of whether an input string is accepted or rejected, except for the case of the PTM, where the final configuration of the tape will be printed.

C. 'a' All steps or nodes will be printed for both deterministic and nondeterministic automata, regardless of whether a solution exists.

13. TRY ANOTHER STRING? (Y OR N)
If 'y' the system will respond:

MAKE ANY CORRECTIONS FIRST? (Y OR N)
Rule 10 above, applies.

14. WANT TO QUIT? (Y OR N)
If 'y' program terminates.
No default.

15. WANT TO STORE PRESENT MACHINE? (Y OR N)
If 'y' the system will respond:

GIVE A NAME TO THIS MACHINE
Name may be any character string of from 1 to 10 symbols in length.
No default.

16. WANT TO RETRIEVE A MACHINE? (Y OR N)
If 'y' the system will respond:

GIVE NAME OF MACHINE
Reply with name of machine. If machine is not in memory, the system will respond:

NO SUCH MACHINE IN MEMORY
Question 16 will be repeated.

If machine is in memory, the system will respond:

(1) WANT RULES DISPLAYED? (Y OR N)
Default 'y'.

(2) GIVE INPUT STRING
No default.

17. WANT TO CREATE A NEW MACHINE? (Y OR N)
If 'y' the system will respond:

TYPE OF MACHINE
Return to rule 1.

If 'n' the terminal session will be terminated.
No default.

SAMPLE TERMINAL SESSION

```
TYPE OF MACHINE?
fa

IS MACHINE DETERMINISTIC? (Y OR N)
y

MAX NUMBER OF STEPS?
25

GIVE RULES

( 1)
(q0,0)-(q0)

( 2)
(q1,0)-(q3)

( 3)
(q2,0)-(q0)

( 4)
(q3,0)-(q1)

( 5)
(q0,1)-(q1)

( 6)
(q1,1)-(q0)

( 7)
(q2,1)-(q3)

( 8)
(q3,1)-(q2)

( 9)
end

ENTER START STATE
q0

ENTER FINAL STATE(S)

:
q0

:
end
```

```
MAKE ANY CORRECTIONS ? (Y OR N)
y

CORRECT ANY RULES? (Y OR N)
y

GIVE RULE NUMBER
1

1 :  (Q0,0)-(Q0)

ENTER CORRECT RULE:
(q0,0)-(q2)

CORRECT ANY MORE RULES? (Y OR N)
n

MAKE ANY MORE CORRECTIONS ? (Y OR N)
n

GIVE INPUT STRING
10010101

WANT A TRACE? (Y,N, OR A)
y
```

STEP	RULE	STATE	STRING
1	0	Q0	10010101
2	5	Q1	0010101
3	2	Q3	010101
4	4	Q1	10101
5	6	Q0	0101
6	1	Q2	101
7	7	Q3	01
8	4	Q1	1
9	6	Q0	

```
STRING IS ACCEPTED

TRY ANOTHER STRING? (Y OR N)
n

WANT TO QUIT? (Y OR N)
n
```

```
WANT TO STORE PRESENT MACHINE? (Y OR N)
y

GIVE A NAME TO THIS MACHINE
even1&0

WANT TO RETRIEVE A MACHINE? (Y OR N)
n

WANT TO CREATE A NEW MACHINE? (Y OR N)
y

TYPE OF MACHINE?
fa

IS MACHINE DETERMINISTIC? (Y OR N)
n

MAX NUMBER OF NODES?
20

GIVE RULES

( 1)
q0,0-q0

( 2)
q0,0-q3

( 3)
q0,1-q0

( 4)
q0,1-q1

( 5)
q1,1-q4

( 6)
q1,0-&

( 7)
q3,0-q4

( 8)
q3,1-&

( 9)
q4,0-q4

(10)
q4,1-q4

(11)
end
```

```
ENTER START STATE
q0

ENTER FINAL STATE(S)

:
q4

:
end

MAKE ANY CORRECTIONS ? (Y OR N)
n

GIVE INPUT STRING
0100

WANT A TRACE? (Y,N, OR A)
y

      STEP    RULE    STATE    STRING

        0       0      Q0       0100

        1       1      Q0       100

        2       3      Q0       00

        3       2      Q3       0

        4       7      Q4

STRING ACCEPTED

NODES =          11

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS FIRST? (Y OR N)
n

GIVE INPUT STRING
0101010110

WANT A TRACE? (Y,N, OR A)
n

MAX NUMBER OF NODES EXCEEDED

WANT TO CHANGE MAXIMUM? (Y OR N)
y

WHAT IS NEW MAX?
50
```

```
STRING ACCEPTED

NODES =         30

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS FIRST? (Y OR N)
n

GIVE INPUT STRING
0100

WANT A TRACE? (Y,N, OR A)
a

      NODE    STATE       INPUT STRING

         0    Q0          0100

         1    Q0          100

         2    Q3          100

         3    Q0          00

         4    Q1          00

         5    &           00

         6    Q0          0

         7    Q3          0

         8    &           0

         9    Q0

        10    Q3

        11    Q4

STRING ACCEPTED

NODES =         11

TRY ANOTHER STRING? (Y OR N)
n

WANT TO QUIT? (Y OR N)
n

WANT TO STORE PRESENT MACHINE? (Y OR N)
y
```

```
GIVE A NAME TO THIS MACHINE
doubles

WANT TO RETRIEVE A MACHINE? (Y OR N)
n

WANT TO CREATE A NEW MACHINE? (Y OR N)
y

TYPE OF MACHINE?
pda

IS MACHINE DETERMINISTIC? (Y OR N)
n

MAX NUMBER OF NODES?
30

GIVE RULES

( 1)
q1,0,r-q1,br

( 2)
q1,1,r-q1,gr

( 3)
q1,0,b-q1,bb

( 4)
q1,0,b-q2,&

( 5)
q1,0,g-q1,bg

( 6)
q1,1,b-q1,gb

( 7)
q1,1,g-q1,gg

( 8)
q1,1,g-q2,&

( 9)
q2,0,b-q2,&

(10)
q2,1,g-q2,&

(11)
q1,&,r-q2,&

(12)
q2,&,r-q2,&
```

```
(13)
end

ENTER START STATE
q1

WHAT IS START SYMBOL?
r

FINAL STATE OR EMPTY STORE? (F OR E)
e

MAKE ANY CORRECTIONS ? (Y OR N)
n

GIVE INPUT STRING
011110

WANT A TRACE? (Y,N, OR A)
y
```

STEP	RULE	STATE	PUSHDOWN	STRING
0	0	Q1	R	011110
1	1	Q1	BR	11110
2	6	Q1	GBR	1110
3	7	Q1	GGBR	110
4	8	Q2	GBR	10
5	10	Q2	BR	0
6	9	Q2	R	
7	12	Q2		

```
STRING ACCEPTED

NODES =         14

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS FIRST? (Y OR N)
n

GIVE INPUT STRING
00011100111000

WANT A TRACE? (Y,N, OR A)
n
```

```
MAX NUMBER OF NODES EXCEEDED

WANT TO CHANGE MAXIMUM? (Y OR N)
y

WHAT IS NEW MAX?
100

STRING ACCEPTED

NODES =         33

TRY ANOTHER STRING? (Y OR N)
n

WANT TO QUIT? (Y OR N)
n

WANT TO STORE PRESENT MACHINE? (Y OR N)
y

GIVE A NAME TO THIS MACHINE
wwr1&0

WANT TO RETRIEVE A MACHINE? (Y OR N)
n

WANT TO CREATE A NEW MACHINE? (Y OR N)
y

TYPE OF MACHINE?
tm

IS MACHINE DETERMINISTIC? (Y OR N)
y

MAX NUMBER OF STEPS?
100

GIVE RULES

( 1)
q1,0-q1,0-0

INCORRECT FORMAT, REPEAT RULE
q1,0-q1,0,r

( 2)
q1,y-q1,y,r

( 3)
q1,1-q2,y,1

( 4)
q2,y-q2,y,1
```

```
( 5)
q2,x-q3,x,r

( 6)
q2,0-q4,0,1

( 7)
q4,0-q4,0,1

( 8)
q4,x-q0,x,r

( 9)
q3,y-q3,y,r

(10)
q3,&-q5,y,r

(11)
q0,0-q1,x,r

(12)
end

ENTER START STATE
q0

ENTER FINAL STATE(S)

:
q5

:
end

MAKE ANY CORRECTIONS ? (Y OR N)
n

GIVE INPUT STRING
0011

WANT A TRACE? (Y,N, OR A)
n

STRING IS ACCEPTED

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS FIRST? (Y OR N)
n

GIVE INPUT STRING
0011
```

```
WANT A TRACE? (Y,N, OR A)
y
```

STEP	RULE	STATE	TAPE
1	0	Q0	->0011
2	11	Q1	X->011
3	1	Q1	X0->11
4	3	Q2	X->0Y1
5	6	Q4	->X0Y1
6	8	Q0	X->0Y1
7	11	Q1	XX->Y1
8	2	Q1	XXY->1
9	3	Q2	XX->YY
10	4	Q2	X->XYY
11	5	Q3	XX->YY
12	9	Q3	XXY->Y
13	9	Q3	XXYY->
14	10	Q5	XXYYY->

```
STRING IS ACCEPTED

TRY ANOTHER STRING? (Y OR N)
y
MAKE ANY CORRECTIONS FIRST? (Y OR N)
n

GIVE INPUT STRING
00011

WANT A TRACE? (Y,N, OR A)
n

STRING NOT ACCEPTED

TRY ANOTHER STRING? (Y OR N)
n

WANT TO QUIT? (Y OR N)
n
```

```
WANT TO STORE PRESENT MACHINE? (Y OR N)
y

GIVE A NAME TO THIS MACHINE
0**n1**n

WANT TO RETRIEVE A MACHINE? (Y OR N)
y

GIVE NAME OF MACHINE
wwr1&0


WANT THE RULES DISPLAYED? (Y OR N)
n


GIVE INPUT STRING
10

WANT A TRACE? (Y,N, OR A)
n

STRING NOT ACCEPTED

NODES =          1

TRY ANOTHER STRING? (Y OR N)
n

WANT TO QUIT? (Y OR N)
n

WANT TO STORE PRESENT MACHINE? (Y OR N)
n

WANT TO RETRIEVE A MACHINE? (Y OR N)
n

WANT TO CREATE A NEW MACHINE? (Y OR N)
y

TYPE OF MACHINE?
ptm

MAX NUMBER OF STEPS?
100

GIVE RULES

( 1)
q0,1-q1,x,r
```

```
( 2)
q1,1-q1,1,r

( 3)
q1,*-q2,*,r

( 4)
q2,y-q2,y,r

( 5)
q2,1-q3,y,r

( 6)
q3,1-q3,1,r

( 7)
q3,*-q4,*,r

( 8)
q4,1-q4,1,r

( 9)
q4,&-q5,1,l

(10)
q5,1-q5,1,l

(11)
q5,*-q5,*,l

(12)
q5,y-q2,y,r

(13)
q2,*-q6,*,l

(14)
q6,y-q6,1,l

(15)
q6,*-q6,*,l

(16)
q6,1-q6,1,l

(17)
q6,x-q0,x,r

(18)
q0,*-q7,*,r

(19)
end
```

```
ENTER START STATE
q0

MAKE ANY CORRECTIONS? (Y OR N)
n

GIVE INPUT STRING
11*11*

WANT A TRACE? (Y,N, OR A)
n

FINAL CONFIGURATION:  XX*->11*1111

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS? (Y OR N)
n

GIVE INPUT STRING
111*11*

WANT A TRACE? (Y,N, OR A)
n

MAX NUMBER OF STEPS EXCEEDED

WANT TO CHANGE MAX? (Y OR N)
y

WHAT IS NEW MAX?
150

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS? (Y OR N)
n

GIVE INPUT STRING
111*11*

WANT A TRACE? (Y,N, OR A)
n

FINAL CONFIGURATION:  XXX*->11*111111

TRY ANOTHER STRING? (Y OR N)
y

MAKE ANY CORRECTIONS? (Y OR N)
n
```

```
GIVE INPUT STRING
1*11*

WANT A TRACE? (Y,N, OR A)
y
```

STEP	RULE	STATE	TAPE
1	0	Q0	->1*11*
2	1	Q1	X->*11*
3	3	Q2	X*->11*
4	5	Q3	X*Y->1*
5	6	Q3	X*Y1->*
6	7	Q4	X*Y1*->
7	9	Q5	X*Y1->*1
8	11	Q5	X*Y->1*1
9	10	Q5	X*->Y1*1
10	12	Q2	X*Y->1*1
11	5	Q3	X*YY->*1
12	7	Q4	X*YY*->1
13	8	Q4	X*YY*1->
14	9	Q5	X*YY*->11
15	10	Q5	X*YY->*11
16	11	Q5	X*Y->Y*11
17	12	Q2	X*YY->*11
18	13	Q6	X*Y->Y*11
19	14	Q6	X*->Y1*11
20	14	Q6	X->*11*11
21	16	Q6	->X*11*11
22	17	Q0	X->*11*11
23	18	Q7	X*->11*11

```
FINAL CONFIGURATION:  X*->11*11
```

```
    /*   AUTO:  AN AUTOMATON SYNTHESIZER                    */

GOLD: PROC OPTIONS(MAIN);

    /*              DECLARATION SECTION

        RULES     -THE ARRAY OF RULES IN CANONICAL FORM
        RULE      -THE ARRAY OF RULES AS INPUT BY THE USER
        STATE     -A STATE OF FINITE CONTROL
        RSTATE    -THE START STATE
        NO        -RULE NUMBER, FOR REPLY STATEMENT
        DUM       -THE START SYMBOL
        LEFT      -THE INITIALIZED LEFT PUSHDOWN STACK
        X,T       -STRINGS USED IN PREPARING OUTPUT IN THE
                   PROPER FORMAT FOR TM AND PTM
        FST       -A FINAL STATE
        FNST      -THE ARRAY OF FINAL STATES
        NUM       -THE PROMPTER FOR RULE INPUT
        MAXUM     -THE CHARACTER REPRESENTATION OF THE
                   MAXIMUM NUMBER OF STEPS OR NODES
        MORE      -OUTPUT STRING CONTAINING EITHER A BLANK
                   OR THE WORD ' MORE '
        FIRT      -OUTPUT STRING CONTAINING EITHER A BLANK
                   OR THE WORD 'FIRST '
        CONFIG    -A MACHINE CONFIGURATION
        INPUT     -INPUT OF RULES
        BUFF      -THE INPUT STRING
        QUE       -SINGLE LETTER REPLY CHARACTERS
        TAPE      -THE TM OR PTM TAPE
        TYPE      -AUTOMATON TYPE
        NAME      -VARIABLE TO CONTAIN NAME OF MACHINE TO BE
                   STORED OR RETRIEVED
        TLPD      -TOP SYMBOL OF LEFT STACK
        TRPD      -TOP SYMBOL OF RIGHT STACK
        TLPDN     -DUPLICATE OF TOP OF LEFT STACK

        LPDA,RPDA
        WRPDA,WLPDA
        RPDOWN,LPDOWN      THESE SIX CHARACTER STRINGS ARE
                           USED TO REPRESENT THE RIGHT AND
                           LEFT PUSHDOWNS; WORKING COPIES OF
                           THESE PUSHDOWNS; AND THE PUSHDOWNS
                           IN THE FORM NEEDED FOR OUTPUT.
        OLD_STATE
        OLD_LFTOP
        OLD_RTTOP    -THE LEFTHAND SIDE OF A CANONICAL RULE;
                      THE STATE, TOP OF THE LEFT STACK, AND
                      THE TOP OF THE RIGHT STACK.  WHEN
                      CONCATENATED TOGETHER, THE STRING
                      REPRESENTS A PARTICULAR CONFIGURATION.
                                                          */


    DCL  RULES(50,4)   CHAR(5),
         STATE  CHAR(3),
         NO   CHAR(2),
         RSTATE CHAR(3),
         RULE(50)  CHAR(50) VARYING,
         DUM    CHAR(1),
         LEFT   CHAR(2),
         X  CHAR(50) VARYING,
         T  CHAR(1),
         FST  CHAR(3),
         NUM    PICTURE 'Z9',
         FNST(10)  CHAR(3),
         MAXUM   CHAR(4),
         (MORE,FIRT)    CHAR(6) VARYING,
         CONFIG   CHAR(5),
```

```
          NAME  CHAR(10),
          TLPD   CHAR(1),
          TRPD   CHAR(1),
          TLPDN  CHAR(1),
          TRPDN  CHAR(1),
          WRPDA  CHAR(100),
          WLPDA  CHAR(100),
          BUFF  CHAR(50),
          QUE    CHAR(1),
          LPDA  CHAR(100) VARYING,
          RPDA  CHAR(100) VARYING,
          INPUT  CHAR(50) VARYING,
          RPDOWN     CHAR(50) VARYING,
          LPDOWN     CHAR(20),
          TAPE       CHAR(50) VARYING,
          TYPE  CHAR(3),
          OLD_STATE    CHAR(3),
          OLD_LFTOP    CHAR(1),
          OLD_RTTOP    CHAR(1);

     /*   STRUCTURE USED TO STORE MACHINES IN MEMORY   */

     DCL  1  STORED BASED(P),
           2  LEFTS  CHAR(2),
           2  NDET,
           2  IPO,
           2  NAMES  CHAR(10),
           2  TYPES  CHAR(3),
           2  RSTATES CHAR(3),
           2  NEMPTYS,
           2  SRULES(50,4) CHAR(5),
           2  SRULE(50)  CHAR(50),
           2  FNSTS(10)  CHAR(3),
           2  NXT PTR;

     /*   POINTERS, FLAGS, & COUNTERS                  */

      DCL TT PTR,
          D PTR,
          E PTR,
          (IGOOD,ITRACE,NODNO,NEMPTY,I,J,NC,NF) FIXED BINARY;


     /*   INITIALIZATION                               */

     IFIRST = 0;
     ON CONVERSION BEGIN;
          DISPLAY(' REPEAT NUMBER') REPLY(MAXUM);
          ONSOURCE = MAXUM;
     END;
GETTING:
     FNST = ' ';
     MORE = ' ';
     FIRT = ' ';
     RULE = ' ';
     RULES = ' ';
     NEMPTY = 0;
     NEW = 0;
     IPRO = 0;
     ICHK = 0;
     QUE = ' ';
     NOND = 0;
     J = 1;
     DISPLAY('TYPE OF MACHINE?') REPLY(TYPE);
     IF TYPE = 'PTM' THEN DO;
          DISPLAY('MAX NUMBER OF STEPS?') REPLY(MAXUM);
          GO TO MAXPTM;
     END;
     IF TYPE = 'FIN' THEN GO TO FINI;
     DISPLAY('IS MACHINE DETERMINISTIC? (Y OR N)')
     REPLY(QUE);
MAXI:
```

```
    IF QUE = 'N' THEN DO;
         NOND = 1;
         DISPLAY('MAX NUMBER OF NODES?')  REPLY(MAXUM);
    END;
    ELSE DISPLAY('MAX NUMBER OF STEPS?') REPLY(MAXUM);
    MAXPTM:
    IF SUBSTR(MAXUM,1,1)<'0' THEN GO TO MAXI;
    MAXNO = MAXUM;

    /*   READ THE RULES IN ONE AT A TIME                    */

    DISPLAY('GIVE RULES');
READ:
    IF NEW = 1 THEN DO;
         NEW = 0;
         J = JTEMP;
         GO TO REPEET;
    END;
    NUM = J;
    DISPLAY('('||NUM||') ') REPLY(INPUT);
    IF INPUT = 'END' THEN GO TO BUILD;
    RULE(J) = INPUT;

    /*   RID THE INPUT STRING OF ALL EMBEDDED BLANKS
         AND PARENTHESIS                                    */

CRUSH:
    M = 1; L = 1;
    NCOM = 0; NDOT = 0;
LOOP:
    DO I = M TO 30 WHILE
    (SUBSTR(INPUT,I,1)¬=' '&SUBSTR(INPUT,I,1)¬='('
    &SUBSTR(INPUT,I,1)¬=')');
         L = L + 1;
    END;
    L = L + 1;
    IF L>30 THEN GO TO TRIM;
    SUBSTR(INPUT,I) = SUBSTR(INPUT,I+1);
    M = I ;
    GO TO LOOP;

    /*   TRIM THE INPUT STRING OF ALL TRAILING BLANKS */

TRIM:
    INPUT = SUBSTR(INPUT,1,INDEX(INPUT,' ')-1);
    IF SUBSTR(INPUT,1,1)<'A' THEN DO;
      ERR:
         DISPLAY('INCORRECT FORMAT, REPEAT RULE') REPLY(INPU
         RULE(J) = ' ';
         RULES(J,*) = ' ';
         RULE(J) = INPUT;
         IF INPUT = 'END' THEN GO TO BUILD;
         GO TO CRUSH;
    END;
    IF INDEX(INPUT,'-')=0 THEN GO TO ERR;
    DO I = 1 TO LENGTH(INPUT);
         IF SUBSTR(INPUT,I,1) = ',' THEN NCOM = NCOM + 1;
         IF SUBSTR(INPUT,I,1) = '-' THEN NDOT = NDOT + 1;
    END;
TRIM1:
    IF TYPE = 'FA ' THEN GO TO F_A;
    IF TYPE = 'PDA' THEN GO TO P_DA;
    IF TYPE = 'PTM' THEN IPRO = 1;
    IF TYPE = 'PTM' THEN TYPE = 'TM ';
    IF TYPE = 'TM ' THEN GO TO T_M;
    DISPLAY('WHAT IS MACHINE?') REPLY(TYPE);
    GO TO TRIM1;

    /*   THE FOLLOWING SECTIONS; F_A, P_DA, AND T_M
         PERFORM THE REQUIRED TRANSFORMATIONS FROM
         THE INPUT RULES TO THE CANONICAL FORM OF
```

```
        THE RULE REQUIRED BY AUTO.  AFTER EACH
        RULE IS TRANSFORMED, A BRANCH IS MADE TO
        THE READ SECTION TO SEE IF THERE ARE ANY
        MORE RULES.  THE INPUT OF RULES IS TERMINATED
        BY INPUTTING 'END'.                              */


F_A:
    IF NCOM ¬= 1 |NDOT¬=1 THEN GO TO ERR;
    OLD_STATE = SUBSTR(INPUT,1,INDEX(INPUT,',')-1);
    OLD_LFTOP = '#';
    OLD_RTTOP = SUBSTR(INPUT,INDEX(INPUT,',')+1,1);
    RULES(J,1) = OLD_STATE||OLD_LFTOP||OLD_RTTOP;
    RULES(J,2) = SUBSTR(INPUT,INDEX(INPUT,'-')+1);
    RULES(J,3) = '#';
    RULES(J,4) = '&' ;
    J = J + 1;
    GO TO READ;
P_DA:
    IF NCOM ¬= 3 |NDOT¬=1 THEN GO TO ERR;
    OLD_STATE = SUBSTR(INPUT,1,INDEX(INPUT,',')-1);
    OLD_LFTOP = SUBSTR(INPUT,INDEX(INPUT,'-')-1,1);
    OLD_RTTOP = SUBSTR(INPUT,INDEX(INPUT,',')+1,1);
    RULES(J,1) = OLD_STATE||OLD_LFTOP||OLD_RTTOP;
    INPUT = SUBSTR(INPUT,INDEX(INPUT,'-')+1);
    RULES(J,2) =  SUBSTR(INPUT,1,INDEX(INPUT,',')-1);
    RULES(J,3) = SUBSTR(INPUT,INDEX(INPUT,',')+1);
    RULES(J,4) = '&' ;
    J = J + 1;
    GO TO READ;
T_M:
    IF NCOM ¬= 3 |NDOT¬=1 THEN GO TO ERR;
    OLD_STATE = SUBSTR(INPUT,1,INDEX(INPUT,',')-1);
    OLD_LFTOP = '#' ;
    OLD_RTTOP = SUBSTR(INPUT,INDEX(INPUT,',')+1,1);
    RULES(J,1) = OLD_STATE||OLD_LFTOP||OLD_RTTOP;
    INPUT = SUBSTR(INPUT,INDEX(INPUT,'-')+1);
    RULES(J,2) =  SUBSTR(INPUT,1,INDEX(INPUT,',')-1);
    INPUT = SUBSTR(INPUT,INDEX(INPUT,',')+1);
    IF SUBSTR(INPUT,INDEX(INPUT,',')+1,1) ='R' THEN BEGIN;
         RULES(J,3) = '#'||SUBSTR(INPUT,1,1) ;
         RULES(J,4) = '&' ;
         J = J+1;
         GO TO READ;
    END;
    IF SUBSTR(INPUT,INDEX(INPUT,',')+1,1) ='L' THEN BEGIN;
         RULES(J,3) = '&&' ;
         RULES(J,4) = '%%'||SUBSTR(INPUT,1,1);
         J = J +1;
         GO TO READ;
    END;
    GO TO ERR;

    /*   THE FOLLOWING SECTIONS; BUILD, FINAL, AND
         GET_STATE CALL FOR INPUTTING OF THE START
         STATE AND IN THE CASE OF THE PDA, THE START
         SYMBOL.  FOR THE PDA, A DETERMINATION IS
         MADE OF WHETHER THE MACHINE IS A FINAL STATE
         OR EMPTY STORE MACHINE.  FOR ALL MACHINES
         WITH FINAL STATE ACCEPTANCE, A LIST OF FINAL
         STATES IS OBTAINED.  THE LEFT PUSHDOWN IS
         INITIALIZED ACCORDING TO THE TYPE OF MACHINE,
         I.E., DELIMITERS PLACED ON THE STACK OR THE
         START SYMBOL FOR THE PDA.                       */


BUILD:
    DISPLAY('ENTER START STATE') REPLY(RSTATE);
    IF RSTATE = '    ' THEN GO TO BUILD;
    IF TYPE = 'FA ' THEN BEGIN;
```

```
          LPDA = '#';
          LEFT = LPDA;
          GO TO FINAL;
     END;
     IF TYPE = 'PDA' THEN BEGIN;
          DUMDUM:
          DISPLAY('WHAT IS START SYMBOL?') REPLY(DUM);
          IF DUM = ' ' THEN GO TO DUMDUM;
          LPDA = DUM||'&';
          LEFT = LPDA;
          FORE:
          DISPLAY('FINAL STATE OR EMPTY STORE? (F OR E)')
          REPLY(QUE);
          IF QUE = 'F' THEN GO TO FINAL;
          IF QUE ¬= 'E' THEN GO TO FORE;
          NEMPTY = 1;
          GO TO REPLAY;
     END;
     IF TYPE = 'TM ' THEN LPDA = '#&';
     LEFT = LPDA;
FINAL:
     IF IPRO = 1 THEN GO TO REPLAY;
     NF = 1;
     DISPLAY('ENTER FINAL STATE(S)');
GET_STATE:
     DISPLAY(': ') REPLY(FST);
     IF FST = 'END' THEN BEGIN;
          IF NFIN = 0 THEN GO TO REPLAY;
          GO TO RESTART;
     END;
     FNST(NF) = FST;
     NF = NF + 1;
     GO TO GET_STATE;


     /*   NEWRULE, REPLAY, REPEET AND REPLAY1-4 HANDLE
          THE INTERACTIVE CHANGES, ADDITONS, DELETIONS,
          ETC. TO THE MACHINE INPUT BY THE USER.        */


NEWRULE:
     DISPLAY('GIVE RULE NUMBER') REPLY(NO);
     IF SUBSTR(NO,1,1)<'0' THEN GO TO NEWRULE;
     KK = NO;
     IF KK<=0|KK>J THEN DO;
          DISPLAY('NO SUCH RULE');
          GO TO REPEET;
     END;
     DISPLAY(NO||':  '||RULE(KK));
     RULE(KK) = ' ';
     RULES(KK,*) = ' ';
     DISPLAY('ENTER CORRECT RULE:  ')  REPLY(INPUT);
     NEW = 1;
     JTEMP = J;
     RULE(KK) = INPUT;
     J = KK;
     GO TO CRUSH;
REPLAY:
     DISPLAY('MAKE ANY'||MORE||'CORRECTIONS'||FIRT||'? (Y OR
     REPLY(QUE);
     IF QUE = 'N' THEN GO TO RESTART;
     IF QUE¬='Y' THEN GO TO REPLAY;
     IF MORE ¬=' ' THEN DO;
          MORE = ' ';
          GO TO REPLAY1;
     END;
REPEET:
     DISPLAY('CORRECT ANY'||MORE||'RULES? (Y OR N)')
     REPLY(QUE);
     IF QUE = 'N' & MORE = ' MORE ' THEN DO;
          FIRT = ' ';
          GO TO REPLAY;
```

```
     END;
     IF QUE = 'N' THEN GO TO REPLAY1;
     IF QUE¬='Y' THEN GO TO REPEET;
     MORE = ' MORE ';
     GO TO NEWRULE;
REPLAY1:
     DISPLAY('ADD ANY RULES? (Y OR N)') REPLY(QUE);
     IF QUE = 'N' THEN GO TO REPLAY2;
     IF QUE ¬= 'Y' THEN GO TO REPLAY1;
     DISPLAY('GIVE RULES');
     GO TO READ;
REPLAY2:
     DISPLAY('CHANGE START STATE? (Y OR N)') REPLY(QUE);
     IF QUE = 'N' THEN GO TO REPLAY3;
     IF QUE ¬= 'Y' THEN GO TO REPLAY2;
     REPLAY21:
     DISPLAY('ENTER START STATE') REPLY(RSTATE);
     IF RSTATE = '    ' THEN GO TO REPLAY21;
REPLAY3:
     DISPLAY('CHANGE FINAL STATE(S)? (Y OR N)') REPLY(QUE);
     IF QUE = 'N' THEN GO TO REPLAY4;
     IF QUE ¬= 'Y' THEN GO TO REPLAY3;
     NFIN = 1;
     GO TO FINAL;
REPLAY4:
     DISPLAY('CHANGE MAX NODES OR STEPS? (Y OR N)')
     REPLY(QUE);
     IF QUE = 'N' THEN GO TO RESTART;
MAX1:
     DISPLAY('WHAT IS NEW MAXIMUM?') REPLY(MAXUM);
     IF SUBSTR(MAXUM,1,1)<'0' THEN GO TO MAX1;
     MAXNO = MAXUM;

     /*  THE RESTART SECTION GETS THE INPUT STRING,
         ADDS DELIMITERS AS NEEDED AND PLACES THE INPUT
         STRING IN THE RIGHT PUSHDOWN.  A CHECK IS
         MADE TO DETERMINE IF A TRACE IS REQUIRED AND
         IF SO, THE APPROPRIATE HEADING  FOR THE
         TYPE OF MACHINE IS PRINTED.                    */


RESTART:
     NODNO = 0;
     DISPLAY('GIVE INPUT STRING') REPLY(BUFF);
     RPDA = BUFF;
     RPDA = SUBSTR(RPDA,1,INDEX(RPDA,' ')-1)||'&';
     RPDA = RPDA||'#';
     LPDA = LEFT;
     STATE = RSTATE;
     ITRACE = 0;
     DISPLAY('WANT A TRACE? (Y,N, OR A)') REPLY(QUE);
     IF QUE = 'Y' THEN ITRACE = 1;
     IF QUE = 'A' THEN ITRACE = 2;
     NC = 0;
     IF ITRACE = 1|(ITRACE = 2 & NOND = 0) THEN DO;
         I = 0;
         IF TYPE = 'FA ' THEN DO;
             DISPLAY('      STEP   RULE   STATE   STRING');
             GO TO MACHINE;
         END;
         IF TYPE = 'PDA' THEN DO;
             DISPLAY('      STEP   RULE   STATE'||
             '  PUSHDOWN              STRING');
             GO TO MACHINE;
         END;
         DISPLAY('      STEP   RULE   STATE    TAPE  ');
     END;

     /*   MACHINE CALLS FOR THE NONDETERMINISTIC
          PROCEDURE WHEN NEEDED, CHECKS THAT THE
```

```
          MAXIMUM NUMBER OF STEPS OR NODES HAS NOT
          BEEN EXCEEDED, AND CAUSES THE STEP-BY-STEP
          OPERATION OF THE AUTOMATON TO BE PRINTED.    */

MACHINE:
     NUM = I;
     IF NOND = 1 THEN BEGIN;
          CALL NONDET;
          IF NODNO<MAXNO THEN DISPLAY('NODES = '||NODNO);
          GO TO RERUN;
     END;
     IF NC = MAXNO THEN DO;
          DISPLAY('MAX NUMBER OF STEPS EXCEEDED');
          DISPLAY('WANT TO CHANGE MAX? (Y OR N)') REPLY(QUE);
          IF QUE = 'N' THEN GO TO RERUN;
     MAX2:
          DISPLAY('WHAT IS NEW MAX?')  REPLY(MAXUM);
          IF SUBSTR(MAXUM,1,1)<'0' THEN GO TO MAX2;
          MAXNO = MAXUM;
     END;
     TLPD = SUBSTR(LPDA,1,1);
     TRPD = SUBSTR(RPDA,1,1);
     IF TRPD = '#' & TYPE = 'TM ' THEN DO;
          TRPD = '&';
          RPDA = '&#';
     END;
     NC = NC + 1;
     IF TLPD = '&'|TRPD = '#' THEN GO TO STUCK;
     IF ITRACE = 1|(ITRACE = 2 & NOND = 0) THEN DO;
         RPDOWN = SUBSTR(RPDA,1,INDEX(RPDA,'&')-1);
          IF TYPE = 'FA ' THEN DO;
               DISPLAY(NC||'      '||NUM||'      '||STATE||
                    '   '||RPDOWN);
               GO TO RUN;
          END;
          IF TYPE = 'PDA' THEN DO;
               LPDOWN = SUBSTR(LPDA,1,INDEX(LPDA,'&')-1);
               DISPLAY(NC||'      '||NUM||'      '||STATE||
                    '   '||LPDOWN||RPDOWN);
               GO TO RUN;
          END;
     IF TYPE = 'TM ' THEN DO;
          ICHK = 1;

     /*   PROD IS A SPECIAL ROUTINE WHICH HANDLES THE
          OUTPUT OF THE TM AND PTM.                    */

PROD:
          X = SUBSTR(LPDA,2,LENGTH(LPDA)-1);
          LEN = LENGTH(X);
          DO I = 1 TO LEN/2;
               T = SUBSTR(X,I,1);
               SUBSTR(X,I,1) = SUBSTR(X,LEN-I+1,1);
               SUBSTR(X,LEN-I+1,1) = T;
          END;
          X = SUBSTR(X,INDEX(X,'&'));
          X = SUBSTR(X,2);
          IF IPRO = 1 & ICHK = 0 THEN BEGIN;
               IF SUBSTR(RPDA,1,1) = '&' THEN RPDOWN= ' ';
               ELSE RPDOWN = SUBSTR(RPDA,1,INDEX(RPDA,'&')-1)
               TAPE = X||'->'||RPDOWN;
               DISPLAY('FINAL CONFIGURATION:  '||TAPE);
               GO TO RERUN;
          END;
          TAPE = X||'->'||RPDOWN;
          DISPLAY
          (NC||'      '||NUM||'      '||STATE||'     '||TAPE);
          IF IPRO = 0 THEN DO;
               DO NR = 1 TO NF;
                    IF STATE = FNST(NR) THEN GO TO POS;
               END;
```

```
        END;
        GO TO RUN;
    END;
    DISPLAY('INCORRECT MACHINE TYPE WAS ENTERED');
    DISPLAY('TYPE ENTERED WAS:  '||TYPE);
    GO TO RERUN;
    END;

    /*   RUN, WITH ITS SUBPROGRAMS, CHANGE AND RIGHT,
         DO THE ACTUAL PROCESSING OF THE STRING IN
         THE DETERMINISTIC MACHINES.  RUN FINDS ANY
         APPLICABLE RULES.  CHANGE, DOES THE NECESSARY
         OPERATIONS ON THE LEFT PUSHDOWN, AND RIGHT
         PUSHDOWN.                                    */

RUN:
    WRPDA = RPDA;
    WLPDA = LPDA;
    CONFIG = STATE||TLPD||TRPD;
    DO I = 1 TO J;
        IF RULES(I,1) = CONFIG THEN GO TO CHANGE;
        IF TYPE = 'PDA' THEN DO;
            IF SUBSTR(RULES(I,1),5,1) = '&' THEN DO;
                IF SUBSTR(RULES(I,1),1,3) = STATE THEN DO;
                    IF SUBSTR(RULES(I,1),4,1) = TLPD THEN
                    GO TO EPSILON;
                END;
            END;
        END;
    END;
    GO TO STUCK;
CHANGE:
    STATE = RULES(I,2);
    IF RULES(I,3) = '& ' THEN DO;
        LPDA = SUBSTR(WLPDA,2);
        GO TO RIGHT;
    END;
    IF RULES(I,3) ='&&' THEN DO;
        LPDA = '#'||SUBSTR(WLPDA,3);
        RPDA=SUBSTR(WLPDA,2,1)||SUBSTR(RULES(I,4),3,1)||
        SUBSTR(WRPDA,2);
        GO TO MACHINE;
    END;
    LPDA = SUBSTR(RULES(I,3),1,INDEX(RULES(I,3),' ')-1)||
    SUBSTR(WLPDA,2);
RIGHT:
    IF RULES(I,4) = '& ' THEN DO;
        RPDA = SUBSTR(WRPDA,2);
        GO TO MACHINE;
    END;
    RPDA = SUBSTR(RULES(I,4),1,INDEX(RULES(I,4),' ')-1)||
    SUBSTR(WRPDA,2);
    GO TO MACHINE;

EPSILON:

    /*   EPSILON TAKES THE REQUIRED ACTION FOR PDA WITH
         EPSILON RULES.

    RPDA = WRPDA;
    STATE = RULES(I,2);
    LPDA = SUBSTR(RULES(I,3),1,INDEX(RULES(I,3),' ')-1)||
    SUBSTR(WLPDA,2);
    GO TO MACHINE;

    /*   WHEN THERE ARE NO LONGER ANY APPLICABLE RULE
         TO APPLY TO A STRING, STUCK IS THE ROUTINE
         WHICH DETERMINES WHETHER THE STRING IS TO
```

```
          BE ACCEPTED OR REJECTED, OR IN THE CASE OF
          THE PTM, THAT THE FINAL CONFIGURATION IS
          TO BE OUTPUT.  THE CHECK FOR ACCEPTANCE IS
          MADE BY CHECKING TO SEE THAT THE ENTIRE
          INPUT STRING HAS BEEN PROCESSED AND THAT THE
          FINITE CONTROL IS IN A FINAL STATE OR FOR THE
          PDA WITH EMPTY STORE, THAT THE LEFT STACK IS
          EMPTY.                                          */

STUCK:
     IF IPRO = 1 THEN DO;
          ICHK = 0;
          GO TO PROD;
     END;
     IF TRPD = '&'|TRPD = '#' THEN DO;
          IF TYPE = 'FA ' THEN GO TO CHKST;
          IF TYPE = 'PDA' THEN DO;
               IF NEMPTY = 0 THEN GO TO CHKST;
               IF TLPD = '&' THEN GO TO POS;
               GO TO NEG;
          END;
CHKST:
          DO I = 1 TO NF;
               IF STATE = FNST(I) THEN GO TO POS;
          END;
     END;
NEG:
     DISPLAY('STRING NOT ACCEPTED');
     GO TO RERUN;
POS:
     DISPLAY('STRING IS ACCEPTED');

     /*   RERUN, NXT_NODE, RETREIVE, LOOK, FILL, AND
          CREATE ARE ROUTINES WHICH ALLOW FOR THE
          RERUNNING OF THE MACHINE, ENDING THE
          TERMINAL SESSION, STORING A MACHINE AWAY IN
          MEMORY, RETREIVING A PREVIOUSLY STORED
          MACHINE, OR CREATING A NEW MACHINE.  IF A
          MACHINE IS RETRIEVED, THE USER HAS THE
          OPTION OF HAVING THE RULES FOR THAT MACHINE
          PRINTED OUT EXACTLY AS THE WERE INPUT.          */

RERUN:
     DISPLAY('TRY ANOTHER STRING? (Y OR N)')  REPLY(QUE);
     IF QUE = 'Y' THEN DO;
          FIRT = ' FIRST';
          GO TO REPLAY;
     END;
     IF QUE ¬='N' THEN GO TO RERUN;
     QUITS:
     DISPLAY('WANT TO QUIT? (Y OR N)')  REPLY(QUE);
     IF QUE = 'Y' THEN GO TO FINI;
     IF QUE ¬= 'N' THEN GO TO QUITS;
     STOW:
     DISPLAY('WANT TO STORE PRESENT MACHINE? (Y OR N)')
     REPLY(QUE);
     IF QUE = 'N' THEN GO TO RETRIEVE;
     IF QUE ¬= 'Y' THEN GO TO STOW;
     NAMIT:
     DISPLAY('GIVE A NAME TO THIS MACHINE') REPLY(NAME);
     IF NAME = ' ' THEN GO TO NAMIT;
     IF IFIRST = 1 THEN GO TO NXT_NODE;
     ALLOCATE STORED SET(FIRST);
          FIRST->SRULES = ' ';
          FIRST->SRULE = ' ';
          FIRST->FNSTS = ' ';
          FIRST->IPO = IPRO;
          FIRST->NAMES = NAME;
          FIRST->TYPES = TYPE;
```

```
          FIRST->RSTATES = RSTATE;
          FIRST->NDET = NOND;
          FIRST->LEFTS = LEFT;
          FIRST->NEMPTYS = NEMPTY;
          FIRST->NXT = NULL;
          DO I = 1 TO 50 WHILE(RULES(I,1)¬='      ');
               DO K = 1 TO 4;
                    FIRST->SRULES(I,K) = RULES(I,K);
               END;
               FIRST->SRULE(I) = RULE(I);
          END;
          DO I = 1 TO NF ;
               FIRST->FNSTS(I) = FNST(I);
          END;
     IFIRST = 1;
     TT = FIRST;
     GO TO RETRIEVE;
NXT_NODE:
     ALLOCATE STORED;
          P->SRULES = ' ';
          P->SRULE= ' ';
          P->FNSTS = ' ';
          TT->NXT = P;
          TT = P;
          P->IPO = IPRO;
          P->NAMES = NAME;
          P->TYPES = TYPE;
          P->NDET = NOND;
          P->LEFTS = LEFT;
          P->RSTATES = RSTATE;
          P->NEMPTYS = NEMPTY;
          P->NXT = NULL;
          DO I = 1 TO 50 WHILE(RULES(I,1)¬='      ');
               DO K = 1 TO 4;
                    P->SRULES(I,K) = RULES(I,K);
               END;
               P->SRULE(I) = RULE(I);
          END;
          DO I = 1 TO NF;
               P->FNSTS(I) = FNST(I);
          END;
RETRIEVE:
     DISPLAY('WANT TO RETRIEVE A MACHINE? (Y OR N)')
     REPLY(QUE);
     IF QUE = 'N' THEN GO TO CREATE;
     IF QUE ¬= 'Y' THEN GO TO RETRIEVE;
     DISPLAY('GIVE NAME OF MACHINE') REPLY(NAME);
     TT = FIRST;
LOOK:
     IF TT->NAMES = NAME THEN GO TO FILL;
     IF TT->NXT = NULL THEN DO;
          DISPLAY('NO SUCH MACHINE IN STORAGE');
          GO TO RETRIEVE;
          END;
     TT = TT->NXT;
     GO TO LOOK;
FILL:
          TYPE = TT->TYPES;
          RSTATE = TT->RSTATES;
          NOND = TT->NDET;
          IPRO = TT->IPO;
          LEFT = TT->LEFTS;
          NEMPTY = TT->NEMPTYS;
          DO I = 1 TO 50 WHILE(TT->SRULES(I,1)¬='      ');
               DO K = 1 TO 4;
                    RULES(I,K) = TT->SRULES(I,K);
               END;
               RULE(I) = TT->SRULE(I);
          END;
          DO N = 1 TO 10 WHILE(TT->FNSTS(N) ¬='    ');
               FNST(N) = TT->FNSTS(N);
          END;
```

```
     DISPLAY('WANT THE RULES DISPLAYED? (Y OR N)')
     REPLY(QUE);
     IF QUE = 'N' THEN GO TO RESTART;
     DO N = 1 TO I-1;
          NUM = N;
          DISPLAY(NUM||': '||RULE(N));
     END;
     GO TO RESTART;
CREATE:
     DISPLAY('WANT TO CREATE A NEW MACHINE? (Y OR N)')
     REPLY(QUE);
     IF QUE = 'Y' THEN GO TO GETTING;
     IF QUE ¬= 'N' THEN GO TO CREATE;


NONDET: PROC;

     /*   NONDET IS THE PROCEDURE WHICH OPERATES ON
          THE NONDETERMINISTIC AUTOMATA.                    */

     IF ITRACE = 2 THEN DO;
          IF TYPE = 'PDA' THEN DISPLAY
          ('       NODE   STATE   PUSHDOWN        '||
          '      INPUT STRING');
          IF TYPE = 'FA ' THEN DISPLAY
          ('       NODE   STATE       INPUT STRING');
          IF TYPE = 'TM ' THEN DISPLAY
          ('       NODE   STATE   TAPE  ');
     END;

     DCL 1 NODE BASED(B),

     /*   NODE IS THE STRUCTURE FOR EACH NODE OF THE
          TREE WHICH IS CONSTRUCTED TO SOLVE THE NON-
          DETERMINISTIC CASE.                               */

             2  S_STATE CHAR(3),
             2  S_PDA  CHAR(200),
             2  S_RULE  PICTURE 'Z9',
             2  POSIT  FIXED BINARY,
             2  NEXT PTR,
             2  UP PTR;
     IGOOD = 0;

     ALLOCATE NODE SET(ROOT);

     /*   THE ROOT NODE IS COMPRISED OF THE START STATE,
          START SYMBOL, IF ANY, AND THE INPUT STRING.
          THIS IS THE INITIAL CONFIGURATION.                */

          ROOT->S_STATE = STATE;
          ROOT->S_PDA = LPDA||RPDA;
          ROOT->S_RULE = 0;
          ROOT->POSIT = LENGTH(LPDA);
          ROOT->NEXT = NULL;
          ROOT->UP = NULL;
     D = ROOT;
     E = D;
     IF ITRACE = 2 THEN DO;
          IF TYPE='PDA' THEN DO;
               LPDOWN = SUBSTR(LPDA,1,1);
               RPDOWN = SUBSTR(RPDA,1,INDEX(RPDA,'&')-1);
               DISPLAY(NODNO||'     '||STATE||'       '||
               LPDOWN||RPDOWN);
          END;
          IF TYPE = 'FA ' THEN DO;
               LPDOWN = ' ';
               RPDOWN = SUBSTR(RPDA,1,INDEX(RPDA,'&')-1);
               DISPLAY(NODNO||'     '||STATE||'         '||RPD
          END;
          IF TYPE = 'TM ' THEN DO;
```

```
            RPDOWN = SUBSTR(RPDA,1,INDEX(RPDA,'&')-1);
            DISPLAY(NODNO||'      '||STATE||'     ->'||RPDOW
        END;
    END;

TOP:

    /*   TOP CHECKS FOR ALL APPLICABLE RULES FOR A GIVEN
         MACHINE CONFIGURATION.  EACH TIME ONE IS FOUND
         THE PROCEDURE CHARGE IS CALLED.  UPON RETURN-
         ING FROM CHARGE EACH TIME, A CHECK IS MADE
         TO DETERMINE WHETHER ACCEPTANCE WAS OBTAINED.
         IF SO, THE PROPER PRINT ROUTINE IS CALLED AS
         REQUIRED.  IF ACCEPTANCE HAS NOT BE OBTAINED
         AND NO APPLICABLE RULES REMAIN FOR A GIVEN
         CONFIGURATION, THEN THE NEXT NODE IS PROCESSED.
         IF NO NEW NODES REMAIN, THEN PROCESSING WILL
         TERMINATE AND A MESSAGE PRINTED.              */

    WLPDA = SUBSTR(D->S_PDA,1,D->POSIT);
    WRPDA = SUBSTR(D->S_PDA,D->POSIT+1);
    TLPD = SUBSTR(WLPDA,1,1);
    TRPD = SUBSTR(WRPDA,1,1);
    CONFIG = D->S_STATE||TLPD||TRPD;
    DO I = 1 TO J;
        IF TYPE¬='PDA' THEN GO TO NOTPDA;
        IF SUBSTR(RULES(I,1),5,1) ='&' THEN DO;
            IF SUBSTR(RULES(I,1),1,3) = D->S_STATE THEN DO
                IF SUBSTR(RULES(I,1),4,1) = TLPD THEN CAL
            END;
        END;
        NOTPDA:
        IF RULES(I,1) = CONFIG THEN BEGIN;
            CALL CHARGE;
            IF TYPE = 'TM '& IPRO = 0 THEN DO;
                DO JJ = 1 TO NF;
                    IF B->S_STATE = FNST(JJ) THEN IGOOD
                END;
            END;
            IF IGOOD = 1 THEN BEGIN;
                IF ITRACE = 1 THEN CALL LINKUP;
                DISPLAY('STRING ACCEPTED');
                RETURN;
            END;
        END;
    END;
    IF D->NEXT= NULL THEN BEGIN;
        DISPLAY('STRING NOT ACCEPTED');
        RETURN;
    END;
    D = D->NEXT;
    IF NODNO = MAXNO THEN DO;
        DISPLAY('MAX NUMBER OF NODES EXCEEDED');
        DISPLAY('WANT TO CHANGE MAXIMUM? (Y OR N)')
        REPLY(QUE);
        IF QUE = 'N' THEN RETURN;
    MAX3:
        DISPLAY('WHAT IS NEW MAX?') REPLY(MAXUM);
        IF SUBSTR(MAXUM,1,1)<'0' THEN GO TO MAX3;
        MAXNO = MAXUM;
    END;
    GO TO TOP;

NONEPS: PROC;

    /*   THE PROCEDURE NONEPS HANDLES THE EPSILON RULES
         FOR NONDETERMINISTIC PUSHDOWN AUTOMATA.       */
```

```
     ALLOCATE NODE;
     NODNO = NODNO + 1;
     B->UP = D;
     E->NEXT = B;
     B->NEXT = NULL;
     E = B;
     B->S_STATE = RULES(I,2);
     B->S_RULE = I;
     RPDA = WRPDA;
     LPDA = SUBSTR(RULES(I,3),1,INDEX(RULES(I,3),' ')-1)||
     SUBSTR(WLPDA,2);
     IF INDEX(LPDA,'&')¬=0 THEN DO;
          LPDA = SUBSTR(LPDA,1,INDEX(LPDA,'&')-1);
     END;
     B->S_PDA = LPDA||RPDA;
     B->POSIT = LENGTH(LPDA);
     IF ITRACE = 2 THEN DO;
          LPDOWN = SUBSTR(LPDA,1,LENGTH(LPDA));
          RPDOWN = SUBSTR(RPDA,1,INDEX(RPDA,'&')-1);
          DISPLAY(NODNO||'     '||B->S_STATE||'     '||
          LPDOWN||RPDOWN);
     END;
END NONEPS;

CHARGE: PROC;

     /*   CHARGE ALLOCATES A NEW NODE FOR EACH APPLICABLE
          RULE FOR SOME PARTICULAR CONFIGURATION.  IT
          DOES THIS IN MUCH THE SAME MANNER AS WAS
          DONE IN THE DETERMINISTIC CASE.    A CHECK IS
          ALSO MADE FOR ACCEPTANCE AT THIS LEVEL.  IF
          THE USER HAD SELECTED TO SEE ALL OF THE NODES
          DISPLAYED AS THEY ARE ALLOCATED, THIS IS THE
          SECTION WHICH WILL CAUSE THE PRINTOUT.       */

     ALLOCATE NODE;
     NODNO = NODNO + 1;
     B->UP = D;
     E->NEXT = B;
     B->NEXT = NULL;
     E = B;
     B->S_STATE = RULES(I,2);
     B->S_RULE = I;
     IF RULES(I,3) = '& ' THEN DO;
          LPDA = SUBSTR(WLPDA,2);
          GO TO CIGHT;
     END;
     IF RULES(I,3) = '&&' THEN DO;
          LPDA = '#'||SUBSTR(WLPDA,3);
          RPDA = SUBSTR(WLPDA,2,1)||SUBSTR(RULES(I,4),3,1)||
          SUBSTR(WRPDA,2);
          GO TO CACHINE;
     END;
     LPDA = SUBSTR(RULES(I,3),1,INDEX(RULES(I,3),' ')-1)||
     SUBSTR(WLPDA,2);
CIGHT:
     IF RULES(I,4) = '& ' THEN DO;
          RPDA = SUBSTR(WRPDA,2);
          GO TO CACHINE;
     END;
     RPDA = SUBSTR(RULES(I,4),1,INDEX(RULES(I,4),' ')-1)||
     SUBSTR(WRPDA,2);
CACHINE:
     B->S_PDA = LPDA||RPDA;
     B->POSIT = LENGTH(LPDA);
     TLPDN = SUBSTR(LPDA,1,1);
     TRPDN = SUBSTR(RPDA,1,1);
     IF TRPDN = '&'|TRPDN= '#' THEN DO;
```

```
        IF TYPE = 'FA ' THEN GO TO CKST;
        IF TYPE = 'PDA' THEN DO;
             IF NEMPTY = 0 THEN GO TO CKST;
             IF TLPDN = '&' THEN IGOOD = 1;
        END;
NODOUT:
    IF ITRACE = 2 THEN DO;
        IF SUBSTR(B->S_PDA,1,1)='&' THEN LPDOWN ='  &';
        IF SUBSTR(B->S_PDA,1,1) =' ' THEN LPDOWN = '  &';
        LPDOWN = SUBSTR(B->S_PDA,1,INDEX(B->S_PDA,'&')-1)
        RPDOWN = SUBSTR(B->S_PDA,B->POSIT+1);
        IF SUBSTR(RPDOWN,1,1)='&'|SUBSTR(RPDOWN,1,1)='#'
        THEN RPDOWN = '  &';
        RPDOWN = SUBSTR(RPDOWN,1,INDEX(RPDOWN,'&')-1);
        IF TYPE = 'FA ' THEN DO;
             LPDOWN = ' ';
             DISPLAY(NODNO||'    '||B->S_STATE||'
             RPDOWN);
             RETURN;
        END;
        IF TYPE = 'PDA' THEN DO;
             DISPLAY(NODNO||'    '||B->S_STATE||'     '||
             LPDOWN||RPDOWN);
             RETURN;
        END;
        IF TYPE = 'TM ' THEN DO;
             X = SUBSTR(LPDOWN,2,INDEX(LPDOWN,' ')-1);
             LEN = LENGTH(X);
             DO L = 1 TO LEN/2;
                  T = SUBSTR(X,L,1);
                  SUBSTR(X,L,1) = SUBSTR(X,LEN-L+1,1);
                  SUBSTR(X,LEN-L+1,1) = T;
             END;
             X = SUBSTR(X,INDEX(X,'&')+1);
             TAPE = X||'->'||RPDOWN;
             DISPLAY(NODNO||'    '||B->S_STATE||'   '||T
             RETURN;
        END;
    END;
CKST:
        DO L = 1 TO NF;
            IF B->S_STATE = FNST(L) THEN IGOOD = 1;
        END;
    END;
    IF ITRACE = 2 THEN GO TO NODOUT;
END CHARGE;

LINKUP: PROC;

    /*   LINKUP IS THE PROCEDURE BY WHICH A PRINTOUT OF
         THE SOLUTION PATH THROUGH THE TREE IS MADE.
         THE TREE IS TRAVERSED FIRST FROM THE SOLUTION
         NODE TO THE ROOT, BY MEANS OF THE POINTERS
         TO THE PREDECESSOR NODE.  AS THE TREE IS
         TRAVERSED IN THIS MANNER, THE POINTERS ARE
         REVERSED UNTIL THE ROOT NODES IS REACHED.  AT
         THIS TIME, THE TREE IS TRAVERSED FROM THE
         ROOT TO THE SOLUTION NODE, PRINTING OUT EACH
         NODE ALONG THE WAY.  THE PRINTOUT WILL BE
         IN THE FORM CORRESPONDING TO THE TYPE OF
         MACHINE WHICH THE USER HAD SPECIFIED.        */

LINK:
    NC = 0;
    D->NEXT = E;
    E = D;
    IF D->UP = NULL THEN GO TO STARTDOWN;
    D = D->UP;
    GO TO LINK;
```

```
STARTDOWN:
     IF TYPE = 'FA ' THEN GO TO FA_DOWN;
     IF TYPE = 'PDA' THEN GO TO PDA_DOWN;
     IF TYPE = 'TM ' THEN GO TO TM_DOWN;
     RETURN;
FA_DOWN:
     RPDOWN = SUBSTR(D->S_PDA,D->POSIT+1);
     RPDOWN = SUBSTR(RPDOWN,1,INDEX(RPDOWN,'&')-1);
     DISPLAY(NC||'      '||D->S_RULE||'      '||D->S_STATE||
     '      '||RPDOWN);
     IF D->NEXT = NULL THEN GO TO OUT;
     D = D->NEXT;
     NC = NC + 1;
     GO TO FA_DOWN;
PDA_DOWN:
     IF SUBSTR(D->S_PDA,1,1) = '&' THEN LPDOWN = '    &';
     IF SUBSTR(D->S_PDA,1,1) = ' ' THEN LPDOWN = '    &';
     LPDOWN = SUBSTR(D->S_PDA,1,INDEX(D->S_PDA,'&')-1);
     RPDOWN = SUBSTR(D->S_PDA,D->POSIT+1);
     IF SUBSTR(RPDOWN,1,1) = '&'|SUBSTR(RPDOWN,1,1) = '#'
     THEN RPDOWN = '    &';
     RPDOWN = SUBSTR(RPDOWN,1,INDEX(RPDOWN,'&')-1);
     DISPLAY(NC||'      '||D->S_RULE||'      '||D->S_STATE||
     '    '||LPDOWN||RPDOWN);
     IF D->NEXT = NULL THEN GO TO OUT;
     D = D->NEXT;
     NC = NC + 1;
     GO TO PDA_DOWN;
TM_DOWN:
     RPDOWN = SUBSTR(D->S_PDA,D->POSIT+1);
     RPDOWN = SUBSTR(RPDOWN,1,INDEX(RPDOWN,'&')-1);
     X = SUBSTR(D->S_PDA,2,D->POSIT);
     LEN = LENGTH(X);
     DO I = 1 TO LEN/2;
          T = SUBSTR(X,I,1);
          SUBSTR(X,I,1) = SUBSTR(X,LEN-I+1,1);
          SUBSTR(X,LEN-I+1,1) = T;
     END;
     X = SUBSTR(X,INDEX(X,'&'));
     X = SUBSTR(X,2);
     TAPE = X||'->'||RPDOWN;
     DISPLAY(NC||'      '||D->S_RULE||'      '||D->S_STATE||
     '    '||TAPE);
     IF D->NEXT = NULL THEN GO TO OUT;
     D = D->NEXT;
     NC = NC + 1;
     GO TO TM_DOWN;
  OUT:
 END LINKUP;
 FINI: END GOLD;
```

BIBLIOGRAPHY

1. Arbib, M. A., Theories of Abstract Automata, Prentice-Hall, Inc., 1969.

2. Hopcroft, J. E. and Ullman, J. D., Formal Languages and Their Relation to Automata, Addison-Wesley Publishing Company, Inc., 1969.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	LTJG R. C. Bolles, USNR, Code 53Bq Department of Mathematics Naval Postgraduate School Monterey, California 93940	25
4.	LCDR Bennett A. Gold, USN 4235 Fulton Avenue Sherman Oaks, California 91403	1

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

REPORT TITLE

AUTO: An Automaton Simulator

DESCRIPTIVE NOTES (Type of report and inclusive dates)

Master's Thesis ; December 1970

AUTHOR(S) (First name, middle initial, last name)

Bennett Alan Gold

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1970	52	2

CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
PROJECT NO	
	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

ABSTRACT

AUTO: An Automaton Simulator, a program implemented under IBM 360/67 TSS, is capable of simulating any user-defined deterministic or nondeterministic finite automaton, pushdown automaton, Turing machine or procedural Turing machine. The simulation is achieved through the use of two pushdown stacks and a single finite control. This approach provides efficient simulation and avoids the programming duplication required to simulate each type of automata separately. Interactive capabilities make the system particularly well suited to the online design, modification, and testing of user-defined automata.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Automata						
Automaton						
Turing Machine						
Automata Simulator